Лекция № 1

Тема 1. Технологии разработки программного обеспечения (ПО).
Часть 1. Традиционные технологии разработки ПО.

Кафедра «Инженерии программного обеспечения»

Подготовила: к.т.н., доцент Фролова Г.А.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Содержание лекции

1.1.Технологии разработки программного обеспечения (ПО).

- 1.1.1. Исторический и современный взгляд на разработку ПО.
- 1.1.2. Типичная схема разработки ПО.

1.2.Традиционные технологии разработки ПО.

- 1.2.1. Водопадная (waterfall) модель.
- 1.2.2. Спиральная (spiral) модель.
- 1.2.3. Инкрементная (*iterative and incremental development*) модель.
- 1.2.4. Макетирование или прототиппирование (prototyping).

1.3. Гибкие технологии разработки ПО. Основные положения Agile Manifesto.

- 1.3.1. Экстремальное программирование (Extreme Programming).
- 1.3.2. Scrum.
- 1.3.3. Microsoft Solutions Framework.
- 1.3.4. Feature Driven Development.

1.4. Сравнение традиционных и гибких технологий разработки ПО.

Рекомендуемая литература

1. Бек К. Экстремальное программирование / К. Бек.– Спб .: Питер , 2002.
2. Мартин Р. Быстрая разработка программ . Принципы , примеры , практика / Р. Мартин, Д. Ньюкирк, Р. Косс – М.: Издательский дом “Вилья мс ”, 2004.
3. Ауер К. Экстремально е программирование : постановка процесса. С первых шагов и до победного конца / К. Ауер, Р. Миллер. – Спб.: Питер , 2004.
4. Бек К. Экстремальное программирование: разработка через тестирование / К. Бек.– Спб.: Питер, 2003.
5. Бек К. Экстремальное программирование: планирование / К. Бек, М. Фаулер– Спб.: Пит ер , 2003.
6. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию / Д. Астелс, Г. Миллер, М. Новак. – М.: Издательский дом “Вильямс ”, 2002.

1.1.Технологии разработки программного обеспечения (ПО)

Технология разработки программного обеспечения (ПО) — система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Технология разработки ПО должна охватывать разнообразные типы программ, включая перечисленные ниже.

Автономное (пример — текстовый редактор):

– устанавливаемое на одиночный компьютер;

– не связанное с другим программным и аппаратным обеспечением.

Встроенное (пример — автомобильный контроллер)***:***

– часть уникального приложения с привлечением аппаратного обеспечения.

Реального времени (пример — программное обеспечение радиолокатора):

– должны выполнять функции в течение малого интервала времени, обычно нескольких микросекунд.

Сетевое (пример — основанная на веб-технологии видеоигра):

– состоит из частей, взаимодействующих через сеть.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1.2. Исторический и современный взгляд на разработку ПО

Разработка ПО. 1950-1980

Этап вплоть до конца 1970х годов можно считать "темными веками" индустрии разработки ПО. Особенностями данного периода ее развития являются:

Общая неразвитость индустрии. Это вызвано несовершенством как технических средств, так и отсутствием теоретического базиса.

Специфическое ПО. ПО является, по сути дела, штучным продуктом, в основном используемым там же, где и велась его разработка, причем основная масса ПО – это научные и инженерные задачи.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Поначалу стандартизации вообще не придавалось никакого значения: написание программ было скорее творчеством, чем четко определенным процессом, и подчинялось принципу "Кодирование–устранение ошибок". Этот подход может быть представлен в виде следующих активностей:

- ✔ Постановка задачи.
- ✔ Ее выполнение до получения требуемого результата.
- ✔ Если результат не удовлетворяет, возврат к первому шагу.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Понятно, что при такой постановке работ получить ответ на вопрос "Сколько нам понадобится времени для создания того-то и того-то" не представляется возможным. Однако, уже к середине 1970х годов появляются внутрикорпоративные стандарты разработки, соответствующие водопадной модели.

Также в 1976 году выходит ставшая классикой книга Брукса "Мифический человеко-месяц", которая не утратила своей актуальности и поныне, и всячески рекомендуется к прочтению.

Разработка ПО. 1980-1996

Данный период отрасли можно охарактеризовать как "осознание того, что так жить дальше нельзя". Аппаратные средства стали доступными как организациям, так и индивидуальным пользователям, что вызвало грандиозное увеличение объема рынка для программного обеспечения. И можно по праву назвать это время эпохой "триумфального шествия бизнес-приложений". ПО стали "потреблять" не только в местах его разработки, что вызывало, ужесточение требований как к самому ПО, так и усложнению процесса разработки.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Чаще всего часто заказчик располагался за сотни миль от места воплощения его идей в жизнь. Более того, разработчики перестали являться специалистами в предметной области заказчика (как это было ранее). Вместе с повышенной сложностью ПО и увеличившимися трудозатратами на его создание эти факторы привели к тому, что на выходе, после водопадной модели, чаще всего заказчик получал совсем не то, что ему нужно, и продукт отправлялся в мусорную корзину, а вместе с ним – и миллионы долларов.

Выходом из подобной ситуации стала инкрементальная и спиральная модели.

Одновременно делается попытка приспособить существующие в промышленности стандарты качества для ИТ-индустрии. На западе – это стандарты ISO, в СССР – ГОСТы.

Разработка ПО. Современность

Современный этап разработки ПО кратко возможно определить как "быстрее, выше, сильнее". Сложность приложений и их объем повысились еще на несколько порядков, так же, как и стоимость разработки. А одной из основных тенденций стала не просто разработка качественного продукта, а возврат инвестиций от него.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Более того, практика показала ограниченность применявшихся ранее инкрементальной и спиральной моделей, и на смену им появилась и была почти повсеместно принята итеративная модель разработки ПО. Стоит отметить, что большинство успешных проектов было создано именно на ее основе.

Итеративная модель была впервые предложена широким массам ИТ-специалистов компанией Rational (сейчас IBM), и является основой ее методологии RUP. Также она лежит в основе MSF (Microsoft Solutions Framework) и борландовской ALM (Application Lifecycle Management).

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В связи с тем, что разработка ПО постоянно усложняется во всех аспектах, обнаружилась потребность давать четкий ответ на вопрос "Может ли эта организация разработать требуемый продукт?" И, как развитие вопроса, возникла необходимость в наборе свойств, признаков, критериев, позволяющих количественно оценить степень зрелости организации, вероятность ее успеха на ниве создания ПО, так сказать. Отсюда берет корни модель CMM (в дальнейшем развившаяся в CMMI), Capability Maturity Model.

Вместе с тем продолжается совершенствование стандартов разработки ПО, организации самого процесса разработки, что нашло отражение в стандартах ISO 90003:2004 и ISO/IEC 15504.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1.2. Типичная схема разработки ПО

Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации [Стандарт IEEE Std 610.12, Глоссарий]

Стандарты жизненного цикла ПО

- ✔ ГОСТ 34.601-90
- ✔ ISO/IEC 12207:1995 (российский аналог — ГОСТ Р ИСО/МЭК 12207-99)

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Стандарт ГОСТ 34.601-90 предусматривает следующие стадии и ***этапы создания автоматизированной системы (АС)***:

1. Формирование требований к АС
   - Обследование объекта и обоснование необходимости создания АС.
   - Формирование требований пользователя к АС.
   - Оформление отчета о выполнении работ и заявки на разработку АС.
2. Разработка концепции АС
   - Изучение объекта.
   - Проведение необходимых научно-исследовательских работ.
   - Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователей.
   - Оформление отчета о проделанной работе.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3. Техническое задание
   - Разработка и утверждение технического задания на создание АС.
4. Эскизный проект
   - Разработка предварительных проектных решений по системе и ее частям.
   - Разработка документации на АС и ее части.
5. Технический проект
   - Разработка проектных решений по системе и ее частям.
   - Разработка документации на АС и ее части.
   - Разработка и оформление документации на поставку комплектующих изделий.
   - Разработка заданий на проектирование в смежных частях проекта.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

6. Рабочая документация
   - Разработка рабочей документации на АС и ее части.
   - Разработка и адаптация программ.
7. Ввод в действие
   - Подготовка объекта автоматизации.
   - Подготовка персонала.
   - Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями).
   - Строительно-монтажные работы.
   - Пусконаладочные работы.
   - Проведение предварительных испытаний.
   - Проведение опытной эксплуатации.
   - Проведение приемочных испытаний.
8. Сопровождение АС.
   - Выполнение работ в соответствии с гарантийными обязательствами.
   - Послегарантийное обслуживание.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Стандартная последовательность шагов разработки ПО.

1. Понять природу и сферу применения предлагаемого продукта.
2. Выбрать процесс разработки и создать план.
3. Собрать требования.
4. Спроектировать и собрать продукт.
5. Выполнить тестирование продукта.
6. Выпустить продукт и обеспечить его сопровождение.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разновидности ***моделей*** процесса разработки ПО:

– водопадная;

– спиральная (итерационная);

– инкрементная;

– макетирование;

– объектно-ориентированное программирование;

– компонентный подход (COM, CORBA);

– CASE-технологии;

– RAD;

– RUP;

– гибкие (agile) технологии: экстремальное программирование (XP), Scrum, TDD, FDD.

1.2.Традиционные технологии разработки ПО

1.2.1. Водопадная (waterfall) модель

Водопадная или каскадная модель жизненного цикла (англ. *waterfall model*) была предложена в 1970 г. Уинстоном Ройсом. Она предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Разработка рассматривается как последовательность этапов, причем переход на следующий, иерархически нижний этап происходит только после полного завершения работ на текущем этапе (рис. 1.1).

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

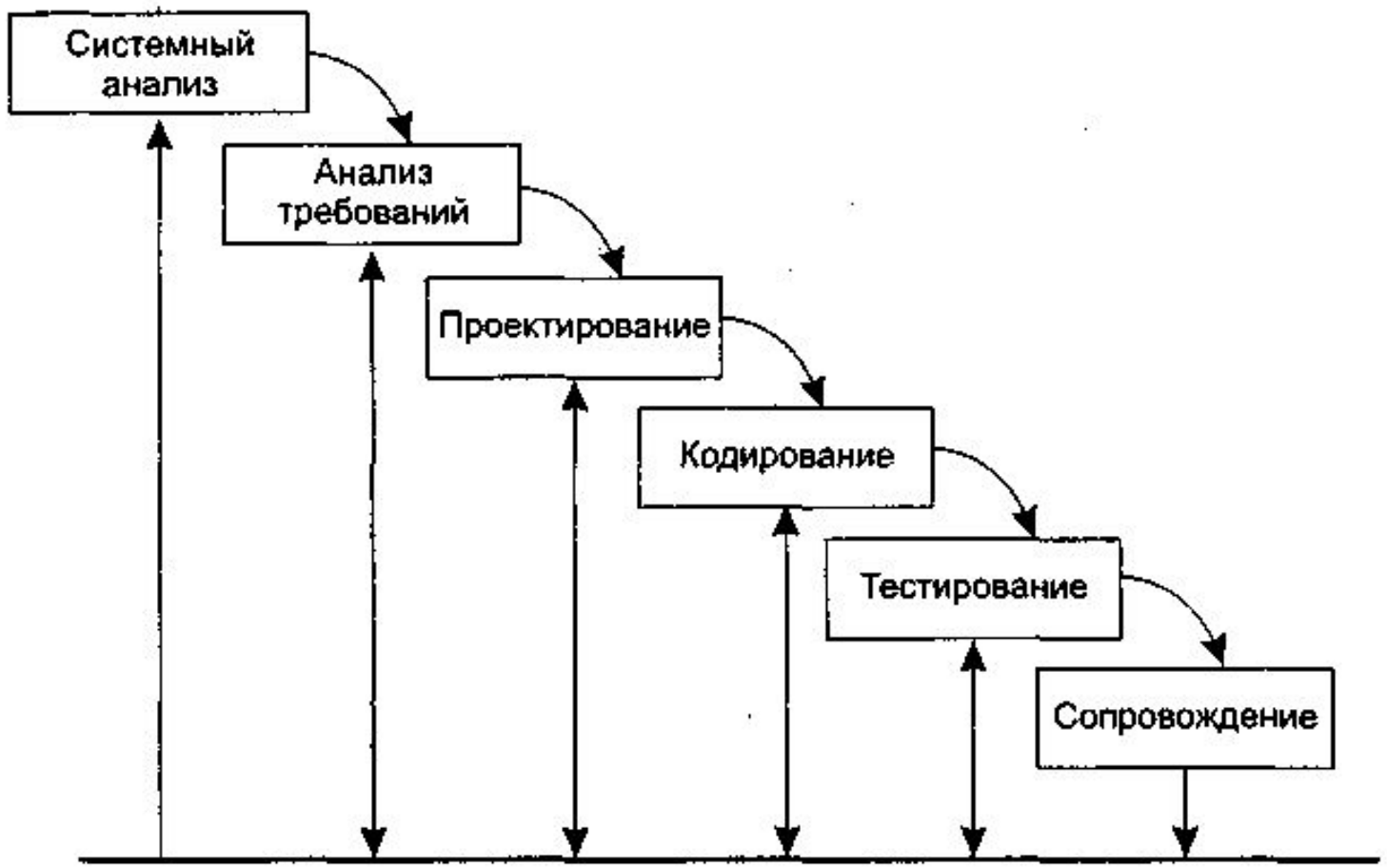


Рис. 1.1. Классический жизненный цикл разработки ПО

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Системный анализ задает роль каждого элемента в компьютерной системе, взаимодействие элементов друг с другом.

Анализ требований относится к программному элементу — программному обеспечению.

Кодирование состоит в переводе результатов проектирования в текст на языке программирования.

Тестирование — выполнение программы для выявления дефектов в функциях, логике и форме реализации программного продукта.

Сопровождение — это внесение изменений в эксплуатируемое ПО.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

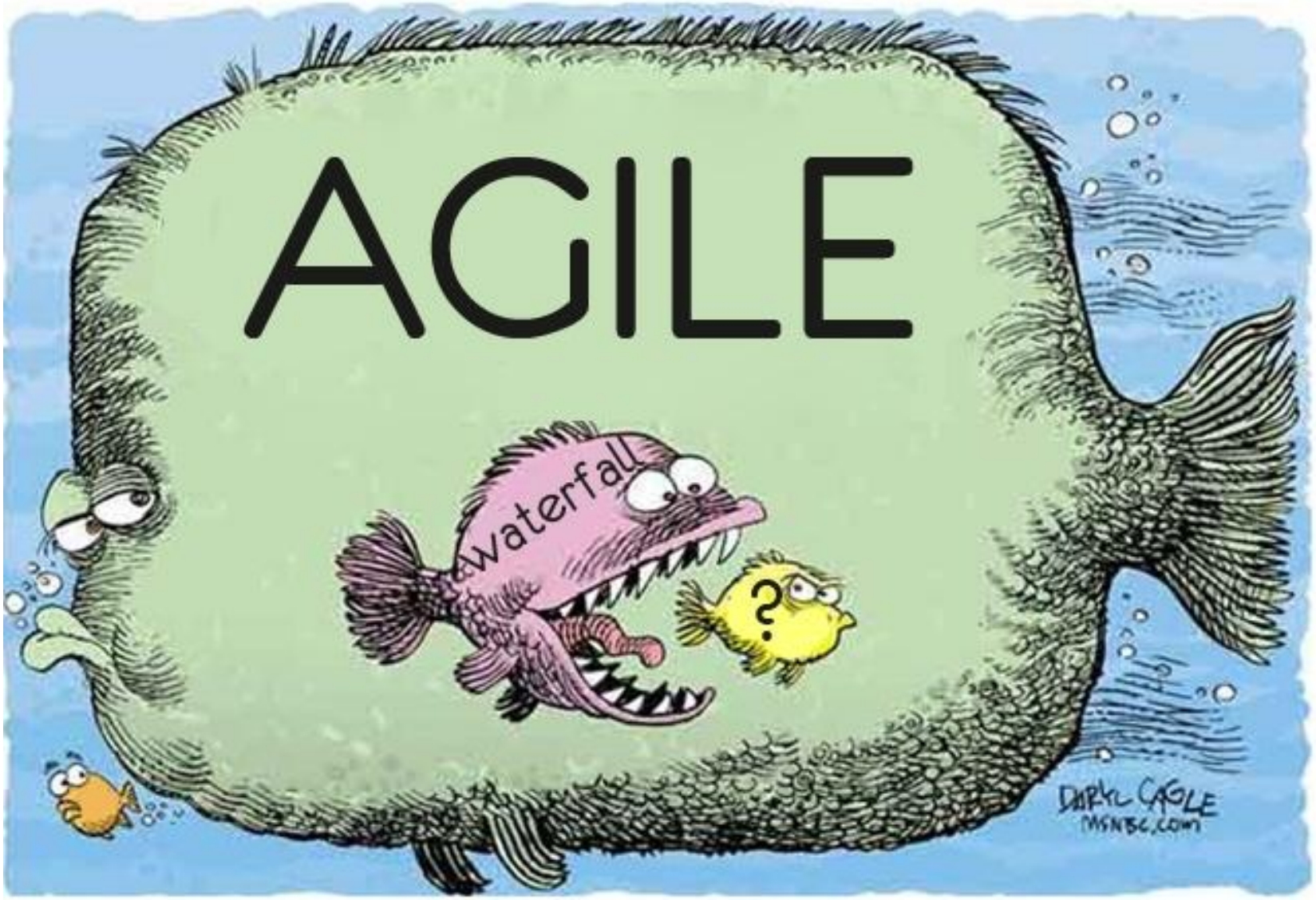
Достоинства водопадной модели:

- ✔ дает план и временной график по всем этапам проекта;
- ✔ упорядочивает ход конструирования;
- ✔ полная и согласованная документация на каждом этапе;
- ✔ легко определить сроки и затраты на проект.

Недостатки водопадной модели:

- ✔ реальные проекты часто требуют отклонения от стандартной последовательности шагов;
- ✔ цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);
- ✔ результаты проекта доступны заказчику только в конце работы.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



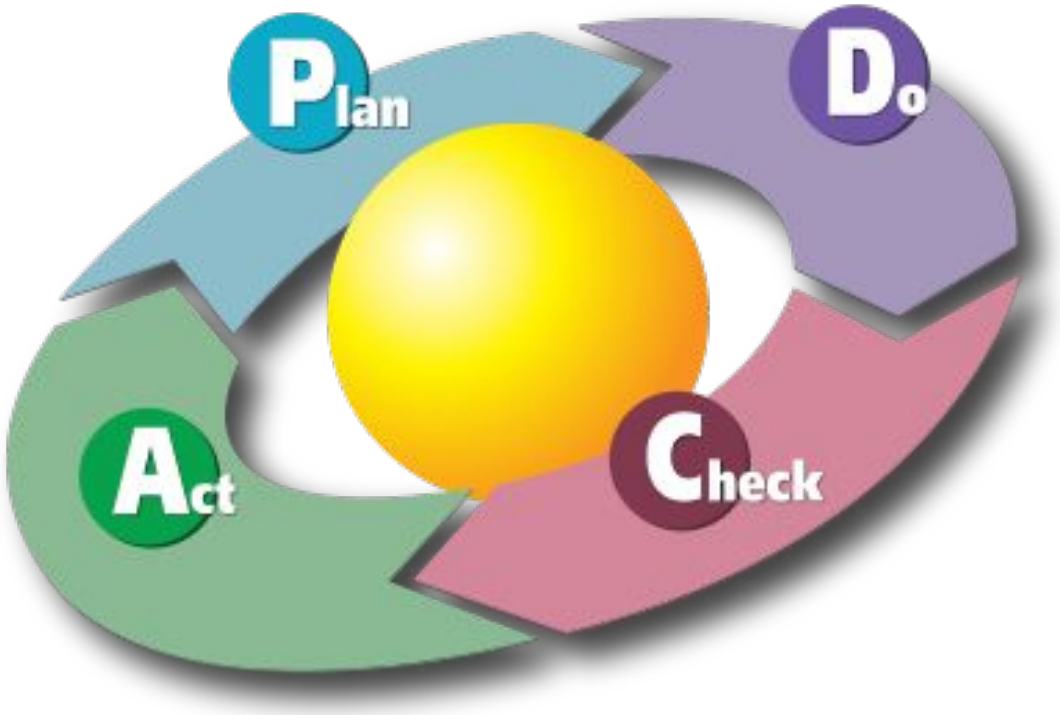
1.2.2. Спиральная (spiral) модель

Спиральная модель (англ. *spiral model*) была разработана в середине 1980-х годов Барри Боэмом. Она основана на классическом цикле Деминга PDCA (plan-do-check-act). При использовании этой модели ПО создается в несколько итераций (витков спирали) методом прототипирования.

Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

PDCA («Plan-Do-Check-Act») циклически повторяющийся процесс принятия решения, используемый в управлении качеством. Также известен как **Deming Cycle**, **Shewhart cycle**, **Deming Wheel**, или **Plan-Do-Study-Act**.



ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методология PDCA представляет собой простейший алгоритм действий руководителя по управлению процессом и достижению его целей. Цикл управления начинается с планирования.

Планирование

установление целей и процессов, необходимых для достижения целей, планирование работ по достижению целей процесса и удовлетворения потребителя, планирование выделения и распределения необходимых ресурсов.

Выполнение

выполнение запланированных работ.

Проверка

сбор информации и контроль результата на основе ключевых показателей эффективности (KPI), получившегося в ходе выполнения процесса, выявление и анализ отклонений, установление причин отклонений.

Воздействие (управление, корректировка)

принятие мер по устранению причин отклонений от запланированного результата, изменения в планировании и распределении ресурсов.

Рис. 1.2. Спиральная модель:
1 — начальный сбор требований и планирование проекта;
2 — та же работа, но на основе рекомендаций заказчика;
3 — анализ риска на основе начальных требований;
4 — анализ риска на основе реакции заказчика;
5 — переход к комплексной системе;
6 — начальный макет системы; 7 — следующий уровень макета;
8 — сконструированная система; 9 — оценивание заказчиком

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Модель определяет четыре действия, представляемые четырьмя квадрантами спирали.

1. Планирование — определение целей, вариантов и ограничений.
2. Анализ риска — анализ вариантов и распознавание/выбор риска.
3. Конструирование — разработка продукта следующего уровня.
4. Оценивание — оценка заказчиком текущих результатов конструирования.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

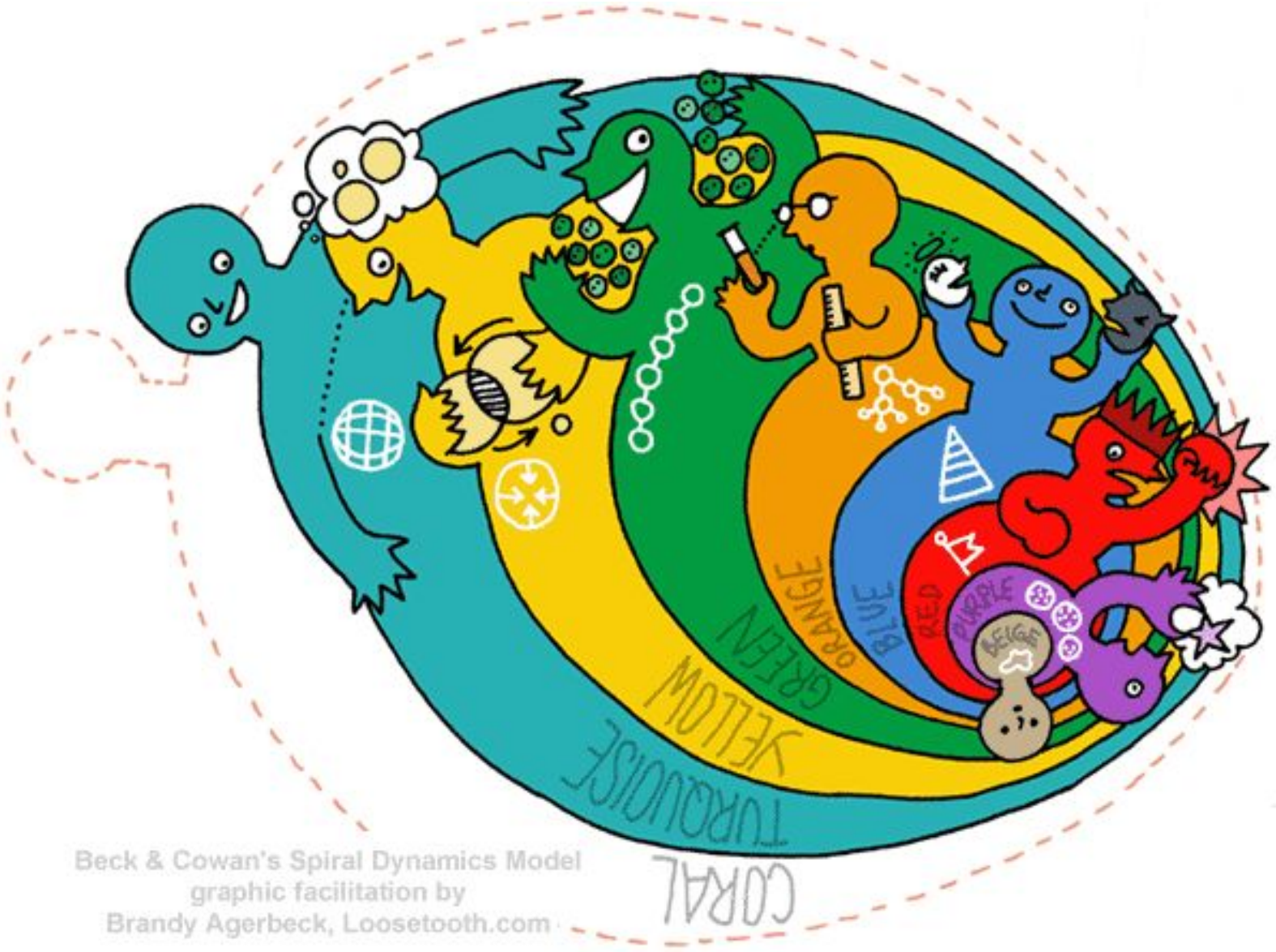
Достоинства спиральной модели:

1) наиболее реально (в виде эволюции) отображает разработку программного обеспечения;
2) позволяет явно учитывать риск на каждом витке эволюции разработки;
3) включает шаг системного подхода в итерационную структуру разработки;
4) использует моделирование для уменьшения риска и совершенствования программного изделия.

Недостатки спиральной модели:

1) новизна (отсутствует достаточная статистика эффективности модели);
2) повышенные требования к заказчику;
3) трудности контроля и управления временем разработки.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



1.2.3. Инкрементная модель

Модель итеративной и инкрементальной разработки (англ. *iterative and incremental development, IID*), получившей также от Т. Гилба в 70-е гг. название *эволюционной модели*.

Модель IID предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом.

Цель каждой *итерации* — получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации. Результат финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации продукт получает приращение — *инкремент* — к его возможностям, которые, следовательно, развиваются *эволюционно*. Итеративность, инкрементальность и эволюционность в данном случае есть выражение одного и то же смысла разными словами со слегка разных точек зрения.

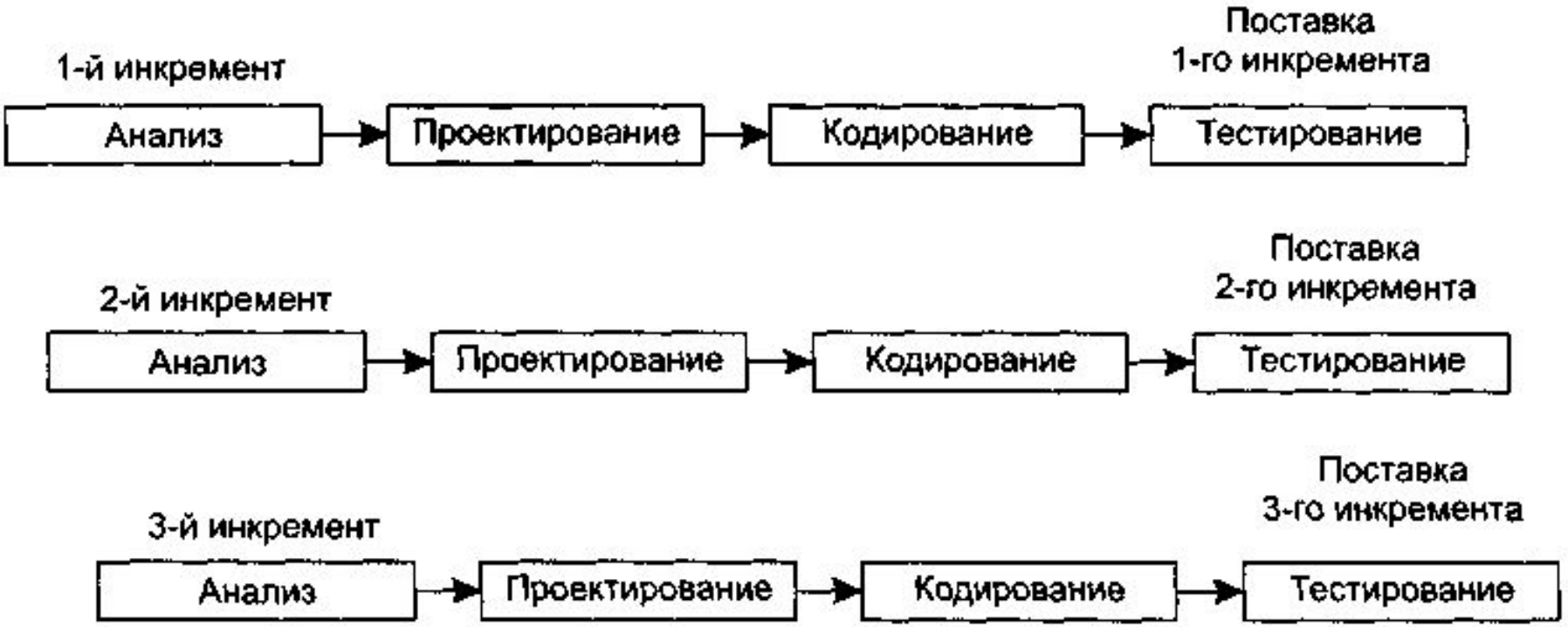


Рис. 1.3. Инкрементная модель

Достоинства итеративной модели:

По выражению Т. Гилба, «эволюция — прием, предназначенный для создания видимости стабильности. Шансы успешного создания сложной системы будут максимальными, если она реализуется в серии небольших шагов и если каждый шаг заключает в себе четко определённый успех, а также возможность «отката» к предыдущему успешному этапу в случае неудачи. Перед тем, как пустить в дело все ресурсы, предназначенные для создания системы, разработчик имеет возможность получать из реального мира сигналы обратной связи и исправлять возможные ошибки в проекте» [Ларман К. Итеративная и инкрементальная разработка: краткая история / К. Ларман, В. Базили // Открытые системы. — 2003.— N 9]

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Недостатки итеративной модели:

1)целостное понимание возможностей и ограничений проекта очень долгое время отсутствует;

2) при итерациях приходится отбрасывать часть сделанной ранее работы;

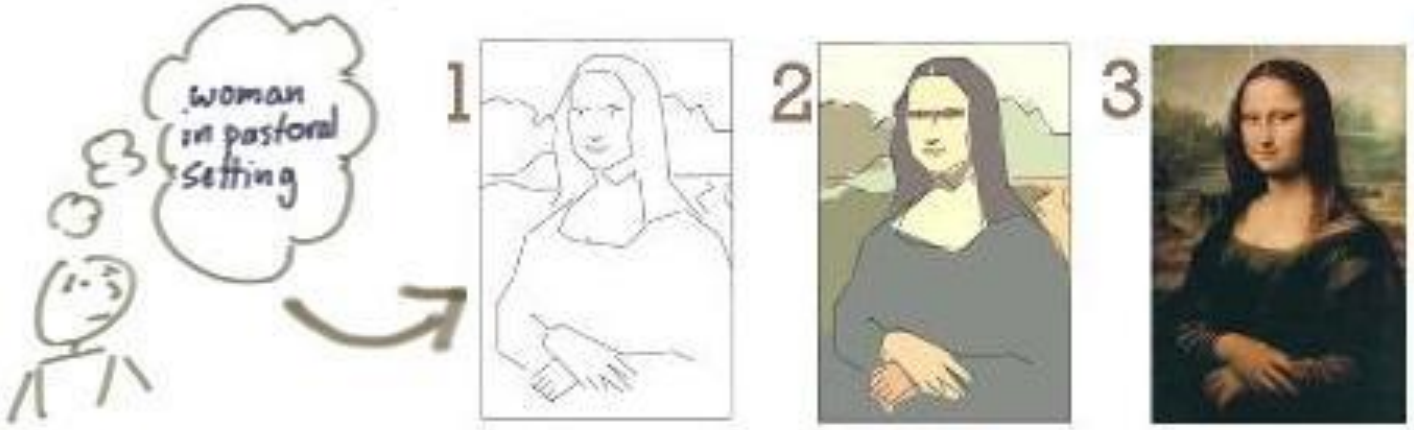
3) добросовестность специалистов при выполнении работ всё же снижается, что психологически объяснимо, ведь над ними постоянно довлеет ощущение, что «всё равно всё можно будет переделать и улучшить позже»;

Различные варианты итерационного подхода реализованы в большинстве современных методологий разработки (RUP, MSF, XP).

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Incremental

Iterative



1.2.4. Макетирование или прототипирование (prototyping)

Прототипи́рование программного обеспечения (от англ. prototyping) — этап разработки программного обеспечения (ПО), процесс создания **прототи́па программы** — макета (черновой, пробной версии) программы, обычно — с целью проверки пригодности предлагаемых для применения концепций, архитектурных и/или технологических решений, а также для представления программы заказчику на ранних стадиях процесса разработки.

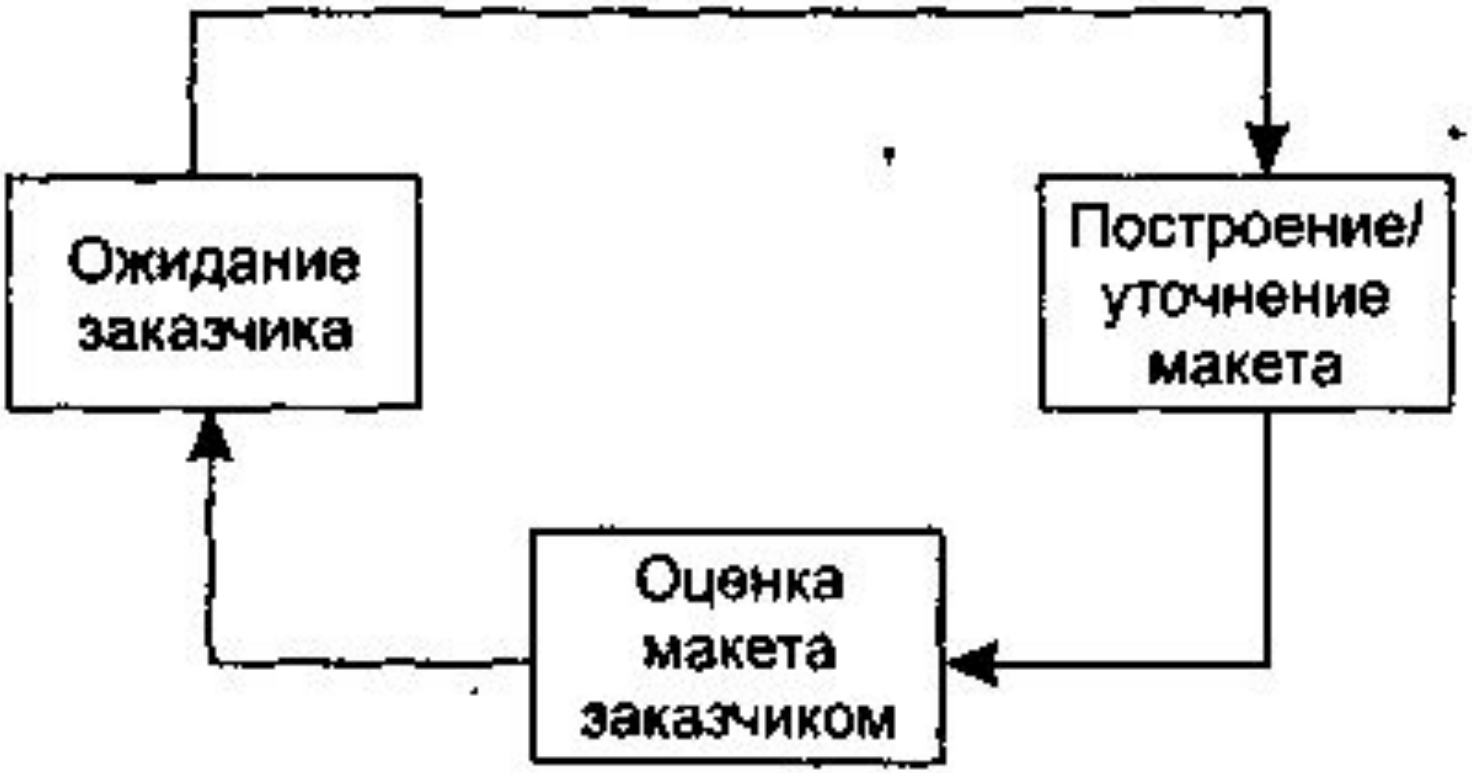


Рис. 1.4. Макетирование

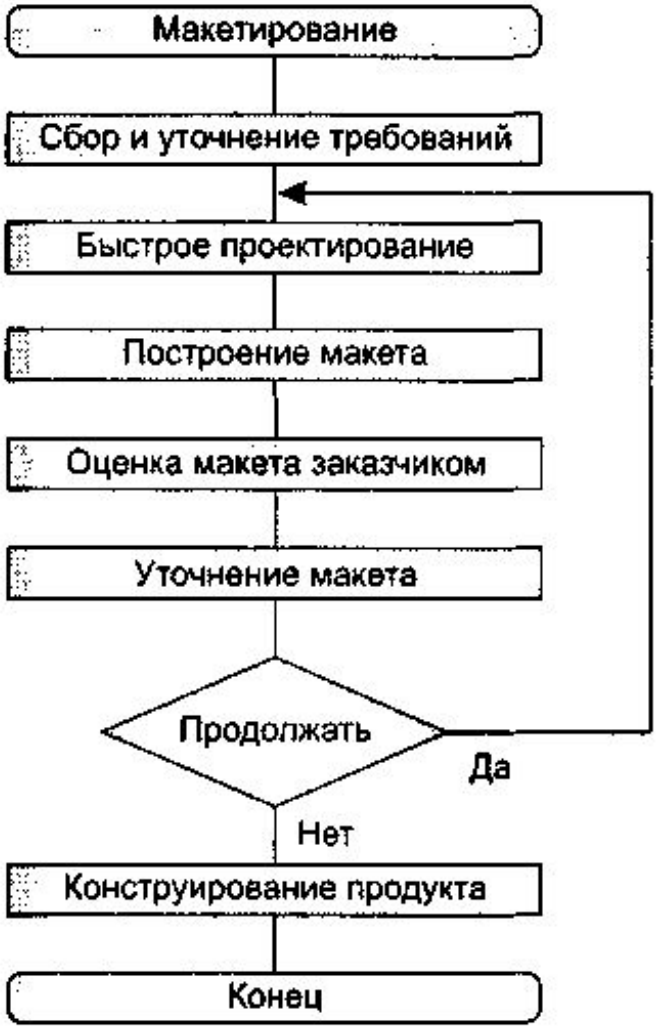


Рис. 1.5. Последовательность действий при макетировании

Быстрое прототипирование

При *быстром прототипировании* (англ. *Rapid rototyping* или *throwaway prototyping*) предполагается, что мы создаем макет, который на каком-то этапе будет оставлен («выброшен») и не станет частью готовой системы.

Эволюционное прототипирование

Эволюционное прототипирование (англ. *evolutionary prototyping*) ставит своей целью последовательно создавать макеты системы, которые будут все ближе и ближе к реальному продукту.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

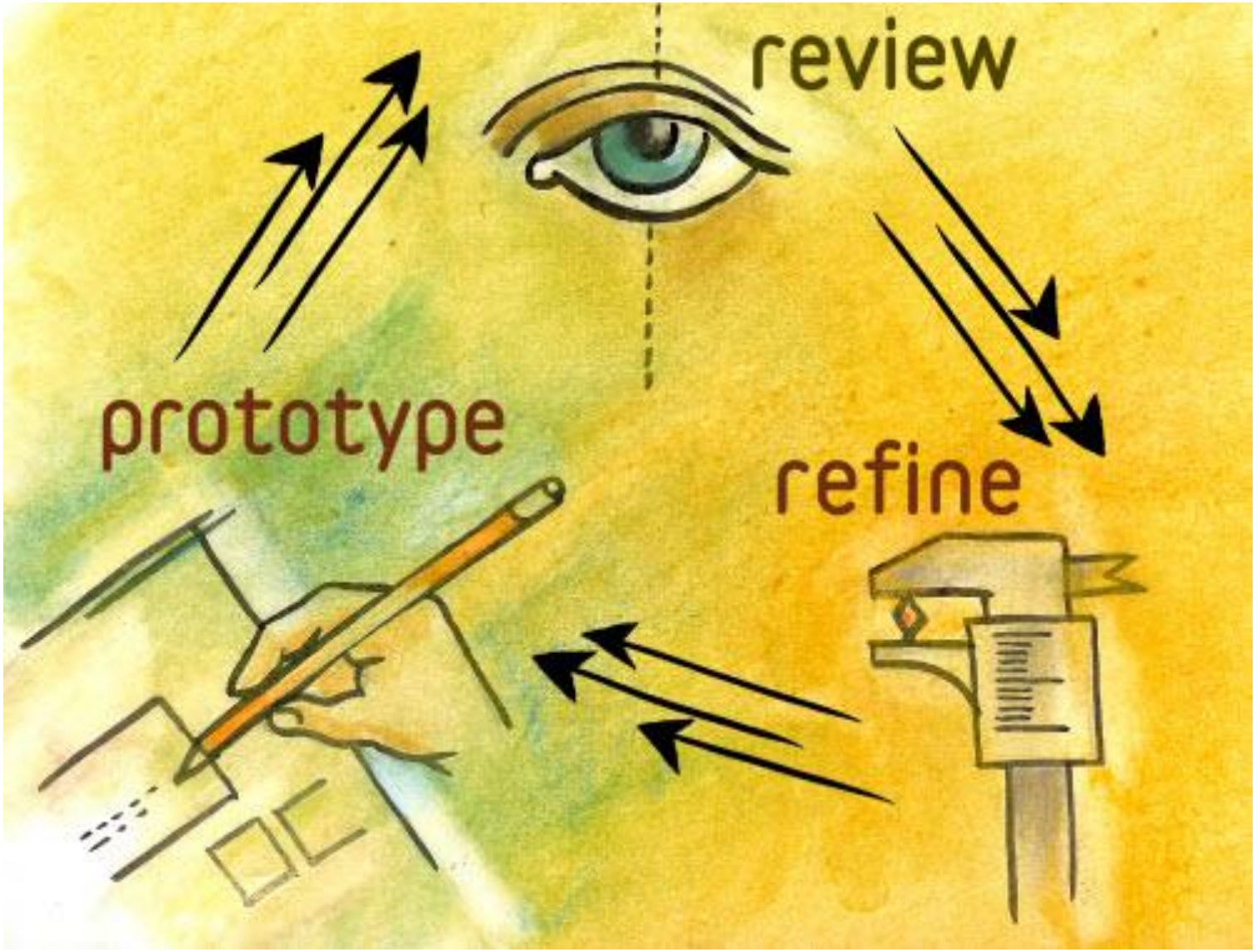
Достоинство прототипирования:

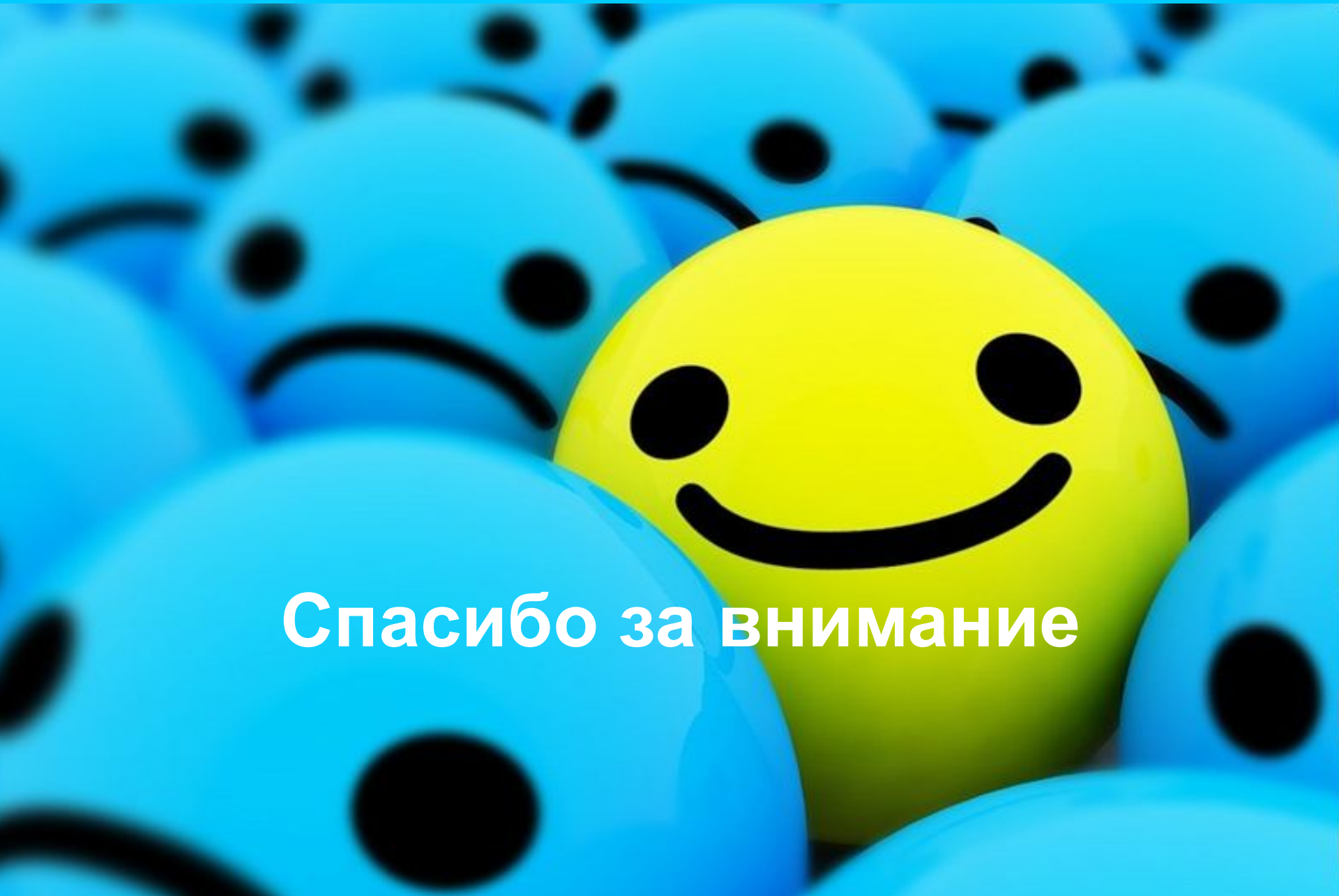
1) обеспечивает определение полных требований к ПО.
2) уменьшение времени, стоимости, рисков;
3)вовлечение пользователя в процесс разработки.

Недостатки прототипирования :

1) заказчик может принять макет за продукт;
2) разработчик может принять макет за продукт;
недостаточный анализ;
3) смешение прототипа и готовой системы в представлении пользователей;
4) чрезмерное время на создание прототипа.

ГИБКИЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ




Спасибо за внимание